Floating-Point Numbers Representation

A binary floating-point number can be represented by

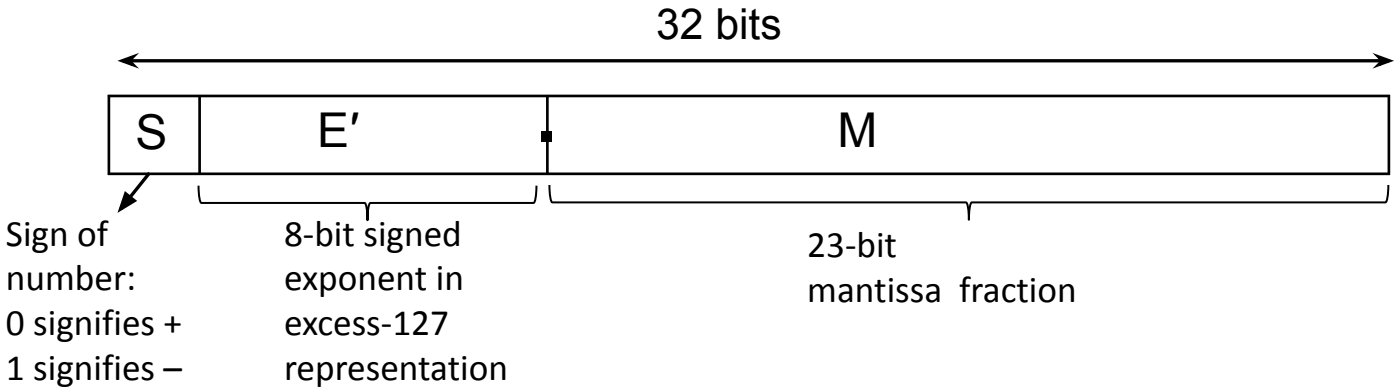
- A sign for the number
- A signed exponent
- Fractional part (mantissa)

There are two IEEE format are used to represent floating point number

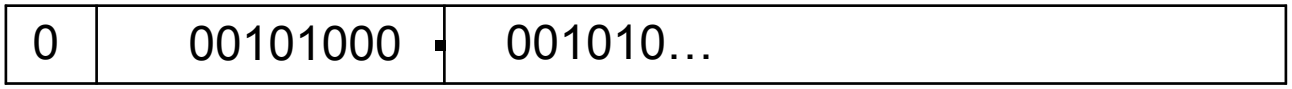
1. Single precision Format (32 bit)
2. Double precision Format (64 bit)

Single precision Format

$E' = E+127$



$$\text{Value represented} = \pm\ 1.M \times 2^{E'-127}$$



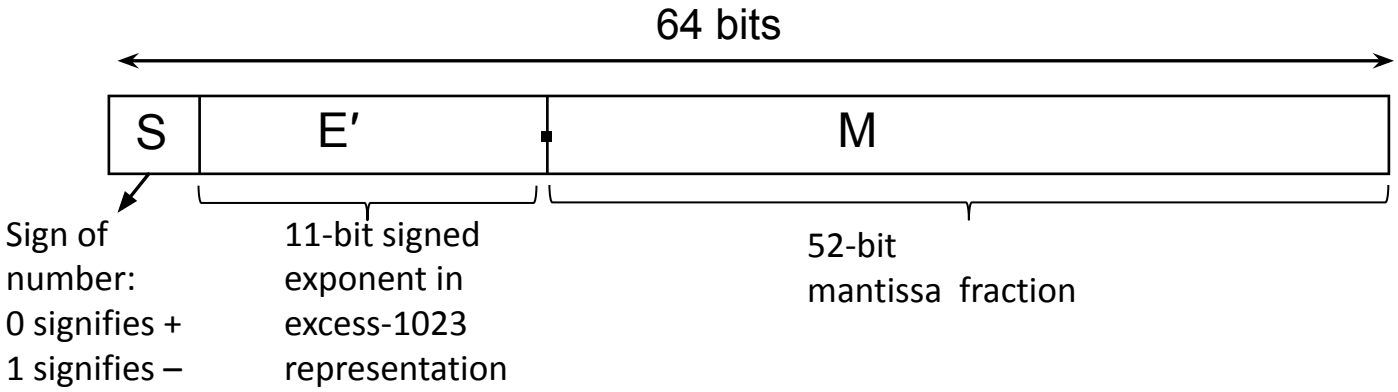
$$\text{Value represented} = +\,1.001010\ldots \times 2^{-87}$$

$E' = -87+127$
$=40$

Single precision Format contd.

- The basic IEEE format for 32-bit representation.
- The leftmost bit represents the sign, S, for the number.
- The next 8 bits, E, represent the signed exponent of the scale factor (with an implied base of 2).
- The remaining 23 bits, M, are the fractional part of the significant bits.
- The full 24-bit string, B, of significant bits, called the mantissa, always has a leading 1, with the binary point immediately to its right.
- Therefore, the mantissa $B = 1.M = 1.b_{-1}b_{-2} \ldots b_{-23}$.
- Instead of the actual signed exponent, E, the value stored in the exponent field is an unsigned integer $E' = E + 127$. This is called the excess-127 format.
- Thus, E is in the range $0 \leq E \leq 255$.
- The end values of this range, 0 and 255, are used to represent special values, as described later.
- Therefore, the range of E' for normal values is $1 \leq E' \leq 254$.
- This means that the actual exponent, E, is in the range $-126 \leq E \leq 127$.

Double precision Format



Value represented = $\pm$ 1.M x $2^{E'-1023}$

1	01100000101	011010…

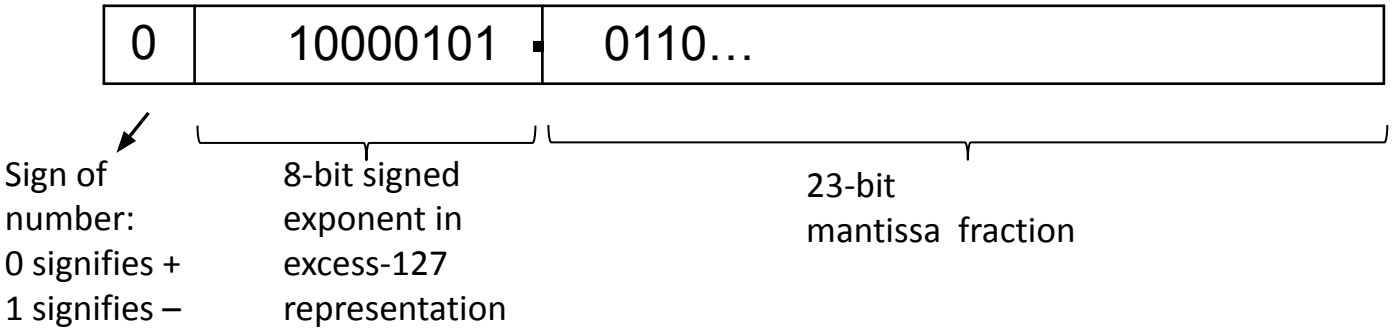
Value represented = - 1.011010… x 2^{-250}

Double precision Format contd.

- The double-precision format has increased exponent and mantissa ranges.
- The leftmost bit represents the sign, S, for the number.
- The 11-bit excess-1023 exponent E has the range $1 \leq E \leq 2046$ for normal values, with 0 and 2047 used to indicate special values as before.
- Thus, the actual exponent E is in the range $-1022 \leq E \leq 1023$.
- The remaining 52 bits, M, are the fractional part of the significant bits.
- The full 53-bit string, B, of significant bits, called the mantissa, always has a leading 1, with the binary point immediately to its right.
- Therefore, the mantissa $B = 1.M = 1.b_{-1}b_{-2} \ldots b_{-52}$.

Normalization

A floating point number is + 0.0010110…….. x 2^{+9}
This is not normalized value.
First, if a number is not normalized, it can be put in normalized form by shifting the binary point and adjusting the exponent.
So, the normalized value is
+1.0110…….. x 2^{+6}



Special Values

The end values 0 and 255 of the excess-127 exponent E' are used to represent special values.

When E' = 0 and M = 0 , the value 0 is represented.

When E' = 0 and M ≠ 0, denormal numbers are represented. There is no implied one to the left of the binary point, and M is any nonzero 23-bit fraction. The purpose of This is useful in dealing with very small numbers, which may be needed in certain situations.

When E' = 255 and M = 0, the value ∞ is represented, where ∞ is the result of dividing a normal number by zero.

When E' = 255 and M ≠ 0, the value represented is called Not a Number (NaN).
A NaN represents the result of performing an invalid operation such as 0/0 or $\sqrt{-1}$.

Example

Represent the decimal value -7.5 in IEEE 754 single precision floating point format.

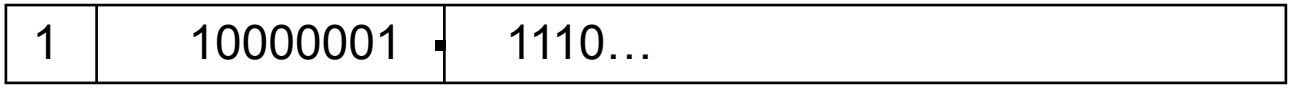
$-(7.5)_{10} = -(111.1)_2 = -1.111 \times 2^2$

The biased exponent $E' = E+127$

$E' = 2+127$

$= 129$

$= (10000001)_2$



Sign of number: 0 signifies + 1 signifies –

8-bit signed exponent in excess-127 representation

23-bit mantissa fraction

Example

Convert -32.75 to IEEE 754 single precision floating point format.

Arithmetic Operations on Floating-Point Numbers

Add/Subtract Rule

1. Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents.
2. Set the exponent of the result equal to the larger exponent.
3. Perform addition/subtraction on the mantissas and determine the sign of the result.
4. Normalize the resulting value, if necessary.

$A = 1.101 \times 2^6$ $B = 1.01 \times 2^4$

$B = .0101 \times 2^6$

$C = 1.1111 \times 2^6$

Arithmetic Operations on Floating-Point Numbers contd.

Multiply Rule

1. Add the exponents and subtract 127 to maintain the excess-127 representation.
2. Multiply the mantissas and determine the sign of the result.
3. Normalize the resulting value, if necessary.

$$E1' + E2' = E1 + 127 + E2 + 127$$

Divide Rule

1. Subtract the exponents and add 127 to maintain the excess-127 representation.
2. Divide the mantissas and determine the sign of the result.
3. Normalize the resulting value, if necessary.

$$E1' - E2' = E1 + 127 - E2 - 127$$

Guard Bits and Truncation

Guard Bits:

Although the mantissas of initial operands and final results are limited to 24 bits, including the implicit leading 1,it is important to retain extra bits, often called guard bits, during the intermediate steps. This yields maximum accuracy in the final results.

Truncation:

There are several ways to truncate the guard bits.

1) Chopping:

Suppose we want to truncate a fraction from six to three bits by this method.

All fractions in the range $0.b_{-1}b_{-2}b_{-3}000$ to $0.b_{-1}b_{-2}b_{-3}111$ are truncated to $0.b_{-1}b_{-2}b_{-3}$.

2) Von Neumann rounding:

If the bits to be removed are all 0s, they are simply dropped, with no changes to the retained bits. However, if any of the bits to be removed are 1, the least significant bit of the retained bits is set to 1. In our 6-bit to 3-bit truncation example, all 6-bit fractions with $b_{-4}b_{-5}b_{-6}$ not equal to 000 are truncated to $0.b_{-1}b_{-2}1$.

3) Rounding :

Rounding achieves the closest approximation to the number being truncated and is an unbiased technique. The procedure is as follows:

$0.b_{-1}b_{-2}b_{-3}1$ . . . is rounded to $0.b_{-1}b_{-2}b_{-3} + 0.001$

$0.b_{-1}b_{-2}b_{-3}0$ . . . is rounded to $0.b_{-1}b_{-2}b_{-3}$.

Example

Use IEEE single precision floating point number to compute 13.25 + 4.5

Thank You